# Control de Congestión Eficiente para Redes HPC con Encaminamiento Adaptativo

Jose Rocher-Gonzalez, Jesus Escudero-Sahuquillo, Pedro J. García y Francisco J. Quiles [1]

*Resumen*— **La red de interconexión es el elemento principal en los clusters de computación de alto rendimiento (HPC) y centros de datos (DC), donde miles de nodos deben comunicarse de forma rápida y fiable. El rendimiento de la red depende de varias opciones de diseño, como la topología, el algoritmo de encaminamiento, la arquitectura del switch, etc. En la literatura se han propuesto algoritmos de encaminamiento altamente eficientes, ya sean deterministas o adaptativos, para equilibrar de forma inteligente los flujos de tráfico dependiendo de la topología de red, pero su rendimiento se reduce en los escenarios en los que la congestión y sus efectos negativos (por ejemplo, el HoL blocking) aparecen. En particular, en escenarios donde la congestión es intensa y persistente, el HoL blocking puede degradar drásticamente el rendimiento de los algoritmos de encaminamiento adaptativo, ya que pueden extender los flujos de tráfico congestionado por todas las rutas disponibles. Además, como hemos demostrado en estudios anteriores, la dispersión de los flujos congestionados puede deteriorar el rendimiento de los esquemas de colas estáticos utilizados para reducir el HoL blocking mediante la separación de los flujos en diferentes colas del switch buffer. De hecho, como estos sistemas se basan en un criterio estático, definido antes de la inyección del tráfico en la red, no pueden evitar que los flujos congestionados y no congestionados compartan colas cuando se combinan con un encaminamiento adaptativo. En este trabajo, proponemos utilizar algunos esquemas de colas estáticos existentes junto a la asignación dinámica de canales virtuales (VC) para aislar en una solo VC los flujos cuyas rutas han sido encaminadas de forma adaptativa, con el fin de evitar que el impacto de la congestión se extienda a través de varias rutas. Básicamente, los flujos adaptados se mueven a un canal especial de flujos adaptados (AFC), de modo que no interactúan con los flujos asignados a otros VC por el esquema de colas estático. De esta manera, se evita el HoL blocking que los flujos adaptados podrían causar a los flujos no adaptados, incluso si los flujos congestionados se han extendido a través de varias rutas. Por otro lado, el esquema de colas estático reducirá sin ninguna interferencia el HoL blocking que puede aparecer entre los flujos no adaptados. Para evaluar nuestra propuesta hemos realizado experimentos de simulación modelando grandes redes de interconexión basadas en la topología Fat-tree. De los resultados obtenidos, podemos concluir que nuestra técnica reduce de manera eficiente y significativa el impacto del HoL-blocking en las redes de interconexión utilizando encaminamiento adaptativo y esquemas de colas cuando aparece la congestión.**

*Palabras clave*— **Redes de Interconexión, Control de Congestión, Esquemas de Colas, Encaminamiento Adaptativo**

## I. Introducción

EN la actualidad, los sistemas de computación de alto rendimiento (HPC) y centros de datos (DC) están compuestos por miles de nodos que trabajan en paralelo para hacer frente a las crecientes demandas, potencia de cálculo y procesamiento de datos, de aplicaciones y servicios utilizados en diversos campos como la genómica, la previsión climática, la inteligencia artificial, la robótica, las redes sociales, etc. En todos estos sistemas, la red de interconexión juega un papel esencial, ya que debe ofrecer un alto ancho de banda y una baja latencia para la comunicación entre los nodos, de lo contrario se convertiría en el cuello de botella del sistema. Para satisfacer las demandas de comunicación, los diseñadores de redes de interconexión se enfrentan a múltiples retos de diseño, como la configuración de topologías de red eficientes y algoritmos de encaminamiento. Concretamente, hay varios requisitos que deben tenerse en cuenta al diseñar la topología de red, como la diversidad de caminos, el bajo diámetro, las condiciones para evitar interbloqueo, la conectividad total, el ancho de banda de la bisección, el coste, etc. En ese sentido, las topologías Fat-Tree [1] se utilizan en sistemas reales de HPC y DC, ya que ofrecen un alto ancho de banda de bisección, diversidad de rutas mínimas y un patrón de conexión que evita los interbloqueos de forma natural. Además, tienen capacidades inherentes de tolerancia a fallos. Gracias a estas características, los Fat-trees se utilizan en grandes supercomputadores como el IBM Summit, el número uno de la última lista TOP500 (noviembre de 2018) desplegando un Fat-tree de tres etapas.

Además, se han propuesto algoritmos de encaminamiento eficientes para los Fat-tree que explotan sus propiedades. Por un lado, los algoritmos deterministas siempre seleccionan la misma ruta entre un par de nodos, pero no tienen en consideración ninguna información sobre el estado de la red o las condiciones de tráfico. Las políticas de encaminamiento deterministas [2], [3] son ampliamente utilizadas porque son fáciles de implementar, y equilibran eficientemente los flujos de tráfico entre las rutas disponibles en la red. Por otro lado, con los algoritmos adaptativos [4], [5], [6] se selecciona la ruta entre dos nodos entre el conjunto de rutas disponibles, y esta selección puede variar dependiendo de las condiciones de tráfico y del estado de la red.

Sin embargo, incluso cuando se utilizan topologías y algoritmos de encaminamiento eficientes, la congestión puede aparecer y degradar severamente el rendimiento de la red. La congestión se produce cuando varios flujos de tráfico solicitan simultáneamente el acceso al mismo recurso de red (por ejemplo, buffers, enlaces, interfaces de red, etc.), ya que se concede a un flujo el acceso al recurso solicitado, mientras que los demás deben esperar. Si esta situación persiste

[1]Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, España. e-mail:{jose.rocher, jesus.escudero, pedrojavier.garcia, francisco.quiles}@uclm.es

en el tiempo, entonces la congestión se propaga por toda la red, llegando finalmente a los nodos fuente, debido a la presión hacia atrás que ejerce el control de flujo.

Esta propagación genera estructuras conocidas como árboles de congestión [7] que consisten en flujos de tráfico congestionados conectados. Los principales efectos negativos de la congestión son el *Head-of-Line(HoL) blocking* [8] y el *buffer hogging* [9].

En consecuencia, se han propuesto muchas técnicas para hacer frente al fenómeno de la congestión y sus efectos negativos. Una de las soluciones se basa en la utilización de varias colas en los puertos de los switches para almacenar diferentes flujos por separado, reduciendo así el HoL blocking. Este enfoque puede llamarse separación de flujos basados en colas. Básicamente, los buffers de los puertos de entrada y/o salida se dividen lógicamente en colas que normalmente se implementan como canales virtuales (VCs) [10] (o carriles virtuales (VLs) en la especificación InfiniBand). Cada cola tiene su propio espacio en el buffer y el control de flujo se realiza a nivel de cola.

Estas técnicas, llamadas esquemas de colas estáticos (SQS), son fáciles de implementar y su eficiencia depende de la política de mapeo específica y del número de colas por puerto. Las políticas de mapeo pueden ser "agnósticas" a la topología y encaminamiento, como VOQnet [11], DAMQs [12] o DBBM [13], o basadas en la topología y encaminamiento, como OBQA [14], vFtree [15] y Flow2SL[16], estas últimas utilizan los recursos de la red de forma más eficiente para reducir el HoL blocking.

Las soluciones descritas para la separación de flujos basadas en colas se proponen, en general, para topologías de red que utilizan encaminamiento determinista. Sin embargo, el encaminamiento adaptativo también es ampliamente utilizado. En este sentido, se realizó un estudio preliminar para analizar el impacto del uso del encaminamiento adaptativo sobre la eficiencia de SQSs inicialmente propuestos para escenarios de encaminamiento determinista [17]. En ese estudio, demostramos que, bajo escenarios de congestión intensa, el encaminamiento oblivious y adaptativo puede extender los árboles de congestión a través de las rutas de red disponibles, de modo que los paquetes congestionados pueden terminar siendo almacenados por la mayoría de las colas definidas por el SQS. Por lo tanto, la probabilidad de HoL blocking aumenta y el rendimiento de la red se degrada. Para evitarlo, propusimos restringir la adaptatividad del encaminamiento mediante umbrales de ocupación en las colas para activar la adaptatividad, de modo que se redujera la propagación de la congestión. Los resultados obtenidos mejoran el comportamiento de la red en situaciones de congestión, pero creemos que el rendimiento de la red puede mejorarse aún más.

Para ello, en este trabajo proponemos una nueva técnica que combina las técnicas de separación de flujos basadas en colas con algoritmos de encaminamiento adaptativo, llamada Aislamiento de Flujo Adaptado (AFI). Básicamente, los flujos de tráfico que no han sido encaminados de forma adaptativa se asignan a un conjunto de colas de acuerdo con un SQS. Sin embargo, cuando la ocupación de una cola supera un determinado umbral, los paquetes almacenados en esta cola se encaminan a través de rutas alternativas y, en el siguiente salto, estos paquetes se almacenan en un VC especial, llamado canal de flujo adaptado (AFC), que no está en el conjunto de colas utilizado por el SQS. Desde este salto, los paquetes adaptados permanecen almacenados en los AFCs para evitar que los flujos congestionados en una ruta alternativa compartan colas con los no adaptados. Además, una vez que un flujo se encamina de forma adaptativa y realiza un salto, se encamina siguiendo rutas deterministas, de modo que se restringe la propagación de la congestión. Una vez que la congestión desaparece, el encaminamiento adaptativo deja de enviar paquetes a través de rutas alternativas.

Hemos evaluado AFI en grandes Fat-trees que conectan hasta 11K nodos, a través de experimentos basados en simulación realizados bajo patrones de tráfico sintéticos y basados en trazas. Los resultados obtenidos muestran que AFI reduce significativamente la probabilidad de que los flujos congestionados y no congestionados compartan colas, así como de que los árboles de congestión se extiendan debido al encaminamiento adaptativo, lo que en general reduce significativamente el HoL blocking en comparación con otros enfoques.

El resto de este documento está organizado de la siguiente manera. La sección II resume los antecedentes en cuanto a la topología Fat-tree, los algoritmos de encaminamiento, la congestión y sus soluciones. En la sección III identificamos los problemas que pueden aparecer cuando se combinan esquemas de colas con el encaminamiento adaptativo. La sección IV describe la técnica de aislamiento de flujo adaptado. En la Sección V, explicamos el modelo de simulación y mostramos y analizamos los resultados obtenidos para evaluar nuestra propuesta.

## II. Conceptos Básicos

### A. Fat-Trees

El Fat-Tree [1] es una topología ampliamente utilizada en los sistemas HPC y DC. Esta familia de topologías multietapa ofrece algunas características deseables y una gran variedad de patrones de conexión. Se han propuesto varios patrones de conexión, como los Real Life Fat-Trees (RLFTs), una subclase de los Parallel Ports Generalized Fat-Trees (PGFTs) [18]. Los RLFTs ofrecen un coste reducido por puerto de switch y, como consecuencia, muchos sistemas comerciales utilizan este patrón de conexión. Hay que tener en cuenta que este patrón de conexión utiliza switches con el mismo número de puertos $P$. En esta topología, cada switch conecta $K$ ($K = P/2$) puertos con otros switches en la siguiente etapa ($K$ es la aridad del switch, es decir, la mitad del número de puertos). El número de nodos $N$ en un RLFT con $n$ etapas y con una aridad de switch $K$ viene dado

por $N = 2(K^n)$, y el número de switches se puede calcular con la fórmula $S = (N \cdot (2n-1))/2K$.

En general, las topologías RLFT tienen las siguientes propiedades:

- Son libres de bloqueos si se utiliza un encaminamiento de ruta mínima, es decir, todas las rutas utilizadas por el algoritmo de encaminamiento ofrecen el menor número de saltos.
- Ofrecen una amplia diversidad de rutas entre cualquier par de nodos, de modo que hay varias rutas de ruta mínima disponibles entre cualquier par de nodos.
- Ofrecen un ancho de banda de bisección constante (CCB) entre todas las etapas.
- Proporcionan de forma inherente capacidad para tolerancia a fallos, debido a la diversidad de caminos.

El encaminamiento en los RLFTs puede dividirse en dos fases: subida y bajada. Primero, en la fase de subida se pueden seleccionar todos los puertos $K$ de un switch, como puertos de salida, para encaminar un paquete determinado a la siguiente etapa. Los paquetes se dirigen en dirección ascendente hasta que llegan a la etapa de giro. Desde la etapa de giro, sólo hay un camino hacia abajo posible para que ese paquete llegue a su destino. Por lo tanto, el conjunto de rutas de recorrido mínimo está limitado por la aridad del switch $K$, y el número de etapas $n$. En consecuencia, la máxima diversidad de rutas de camino mínimo entre cualquier par de nodos es $K^{n-1}$.

Existen varios algoritmos de encaminamiento propuestos para los Fat-trees, ya sean deterministas o adaptativos. Por un lado, existen algoritmos de encaminamiento determinista que aprovechan la diversidad de rutas, como DESTRO (Deterministic destination and STage based Routing) [2] o $D$-mod-$K$ [3], que reparte inteligentemente los flujos entre los caminos disponibles de subida.

Sin embargo, el encaminamiento determinista obliga a los paquetes a seguir siempre la misma ruta entre un par de nodos dado. Esto hace que los algoritmos de encaminamiento determinista funcionen eficientemente cuando se generan patrones de tráfico uniformes en la red (es decir, todos los nodos generan una cantidad similar de tráfico que se distribuye entre los nodos de una manera justa). Por otro lado, el encaminado adaptativo es particularmente útil para equilibrar la carga cuando el trafico no es uniforme y existen propuestas como [4], [5], [6] para implementarlo en los Fat-trees.

La Figura 1 muestra las posibles rutas de camino mínimo entre dos nodos en un RLFT de 3 niveles. Las flechas en negrita representan la ruta utilizada por el encaminamiento determinista $D$-mod-$K$. Las líneas discontinuas muestran las rutas disponibles utilizadas por los algoritmos de encaminamiento adaptativo, además de la ruta determinista, que también puede ser seleccionada por los algoritmos de encaminamiento adaptativo.

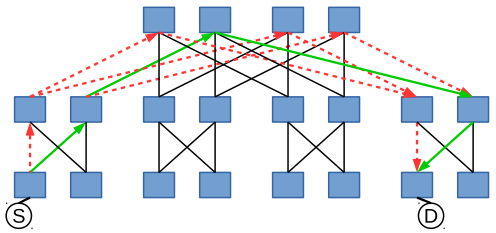



Fig. 1: Rutas posibles entre dos nodos en un RLFT de 3 etapas (k=2). Las líneas verdes representan la ruta $D$-mod-$K$. Las líneas rojas representan las rutas del encaminamiento adaptativo.

### *B. La congestión y sus efectos*

La congestión de la red aparece por varias causas, tales como el tráfico Hotspot, ráfagas de tráfico, nodos populares que normalmente reciben mucho tráfico, etc. Independientemente de la causa, la congestión termina ralentizando los flujos de tráfico, reduciendo drásticamente el rendimiento de la red. Específicamente, el origen de la congestión es la contención, que ocurre dentro de un switch cuando varios flujos, desde diferentes puertos de entrada, solicitan de forma simultánea acceso al mismo puerto de salida. Cuando la contención persiste en el tiempo, entonces aparece la congestión, así como sus efectos negativos.

El problema más importante derivado de la congestión es el HoL blocking [8], que en general aparece cuando un paquete congestionado en la cabeza de una cola impide que otros paquetes almacenados detrás avancen, incluso si solicitan puertos disponibles dentro de un switch. Cuando las colas que contienen paquetes congestionados en su cabecera se llenan, la congestión se propaga a otros switches debido a la presión hacia atrás del control de flujo [1] y pueden llegar a cualquier puerto de la red, formando lo que se conoce como un árbol de congestión. Específicamente, la raíz del árbol de congestión se origina en el puerto de salida del switch congestionado y la congestión se propaga desde la raíz hasta las hojas (hasta el NIC del emisor). Apreciamos diferentes tipos de HoL blocking dependiendo del lugar donde se origina la congestión. Nos referimos a low-order HoL blocking [19] cuando los paquetes se bloquean en el mismo switch donde se origina la congestión. Nos referimos a high-order HoL blocking cuando los paquetes se bloquean en un switch diferente del switch donde se origina la congestión, debido a la presión hacia atrás del control de flujo.

Además, la congestión puede producir buffer hogging [9] cuando los flujos congestionados ocupan la mayor parte del espacio de buffer disponible en cualquier puerto de switch, de tal modo que impiden que otros flujos lleguen a ese puerto. También podemos distinguir diferentes tipos de buffer hogging, como el inter buffer hogging o el intra buffer hogging. El in-

[1] Asumimos redes sin pérdidas, donde no se permite descartar paquetes. Estas redes utilizan un control de flujo a nivel de enlace basado en créditos que evita descartar paquetes cuando los buffers que se solicitan están llenos y su posterior reenvío

ter buffer hogging se produce cuando la ocupación de una cola aumenta hasta consumir todo el espacio de buffer compartido entre colas, mientras que el intra buffer hogging se produce cuando un flujo consume todo el espacio asignado a una cola determinada, impidiendo que otros flujos se almacenen en esa cola. En resumen, el HoL blocking y el buffer hogging pueden degradar severamente el rendimiento de la red de interconexión, a menos que se tomen algunas contramedidas.

### C. Esquemas de colas

Una de las soluciones tradicionalmente ofrecidas para tratar la congestión es la introducción de innovaciones en la arquitectura del switch para mitigar el HoL blocking y el buffer hogging. Una de estas innovaciones consiste en dividir el espacio del buffer de los puertos de switch en canales virtuales (VCs). Aunque los VCs fueron introducidos originalmente para evitar bloqueos, también pueden ser usados para proporcionar calidad de servicio (QoS) [20] o reducir el HoL blocking [15]. En algunos casos, los buffers de los switches pueden implementar *Virtual Output Queues* (VOQs) [12] a nivel físico, por medio de entradas demultiplexadas en el crossbar interno del switch. De hecho, los switches con VOQs evitan el low-order HoL blocking de forma nativa porque almacenan paquetes dirigidos a diferentes puertos de salida en diferentes VOQs, y los paquetes bloqueados almacenados en una VOQ no retrasan el avance de los paquetes dirigidos a otras VOQs diferentes. Sin embargo, los switches basados en VOQ todavía sufren de high-order HoL blocking. Nótese que los switches basados en VOQ también pueden utilizar esquemas de colas (mediante VCs) de forma ortogonal a los VOQs, con el fin de tratar el HoL blocking de forma más eficiente [21]. También hay que tener en cuenta que el control de flujo en los switches basados en VOQ se realiza a nivel VC, de modo que no es posible controlar la ocupación de una VOQ que está ocupando todo el espacio del buffer, debido a una situación de congestión. Por lo tanto, se necesitan otras estrategias para eliminar el buffer hogging y el HoL blocking.

El uso de los esquemas de colas estáticos (SQSs) ha sido ampliamente estudiado porque son alternativas rentables para reducir el HoL blocking y el buffer hogging. Como se ha mencionado anteriormente, los esquemas de colas almacenan diferentes flujos de paquetes en diferentes colas (o VCs) en los puertos del switch, de acuerdo con una política específica de mapeo de flujo a cola. Esta política de mapeo se define como *estática* ya que se aplica antes de que los paquetes se inyecten en la red, y sin considerar ninguna información sobre el estado de la red. La calidad de la política de mapeo puede ser medida [16], y obviamente determina la eficiencia del esquema de colas. Basándose en esta idea, existen esquemas de colas basados en la topología y el encaminamiento, ya que la calidad de su política de mapeo es alta y coincide con la topología y las propiedades de encaminamiento. Por el contrario, hay esquemas de colas agnósticos a la topología y al encaminamiento que no prestan atención a estas propiedades.

En este trabajo nos centramos en tres SQSs que pueden ser utilizados en Real Life Fat-Trees(RLFTs): *Destination-Based Buffer Management* (DBBM) [13], Flow2SL [16] y vFtree [15]. Concretamente, DBBM es un esquema agnóstico a la topología y al encaminamiento que asigna paquetes con destinos consecutivos a diferentes colas en cualquier puerto de switch, mientras que los paquetes con el mismo destino son almacenados en la misma cola. La política de mapeo de DBBM se define con un función módulo $D \bmod Q$, donde $D$ es el ID destino del paquete y $Q$ es el número de colas (o VCs) por puerto. Como tenemos un número limitado de colas (o VCs), DBBM termina almacenando paquetes para un subconjunto de destinos en cada cola. Téngase en cuenta que la política de asignación de DBBM sólo se basa en el ID destino del paquete para seleccionar una cola.

Por otro lado, tanto vFtree como Flow2SL son esquemas que tienen en cuenta la topología y el encaminamiento, especialmente diseñados para Fat-trees que utilizan algoritmos de encaminamiento deterministas, como "$D$-mod-$K$"' [2], [3]. Básicamente, vFtree [15] observa los switches hoja de origen y destino de cada paquete, y baraja los paquetes entre las $Q$ colas disponibles. vFtree almacena en la misma cola los flujos de paquetes que tienen idéntico switch hoja origen y están dirigidos al mismo switch hoja destino, mientras que otros paquetes que tienen el mismo switch hoja origen pero que están dirigidos a un switch hoja destino diferente se barajan entre las $Q$ colas disponibles. Téngase en cuenta que los paquetes dirigidos a switches hoja destino consecutivos se almacenan en colas (o VCs) consecutivas. Desafortunadamente, la política de mapeo de vFtree hace que los paquetes dirigidos a diferentes switches hojas compartan colas en Fat-trees con más de dos etapas. Por lo tanto, para superar este defecto se propuso la técnica Flow2SL [16]. Básicamente, Flow2SL divide la red en grupos de nodos consecutivos y hay tantos grupos como número de colas disponibles ($Q$). De este modo asigna paquetes a diferentes colas dependiendo de su grupo origen y destino. En otras palabras, los movimientos que tienen lugar entre el mismo grupo fuente y el mismo grupo de destino se asignan a la misma cola (o VC), mientras que los movimientos que tienen entre el mismo grupo origen pero se dirigen a grupos destino diferentes se asignan a colas diferentes. Tenga en cuenta que el HoL blocking entre los flujos asignados a la misma cola se reduce a medida que el número de flujos que comparten colas disminuye a través de los niveles del Fat-tree, debido al balanceo de destinos que ofrece el algoritmo de encaminamiento.

El problema de los esquemas de colas propuestos para reducir el HoL blocking es que no reaccionan bien cuando el algoritmo de encaminamiento es adaptativo, es decir, cuando los flujos pueden ser encaminados a través de todas las rutas disponibles. En este

caso, cuando aparece la congestión, el algoritmo de encaminamiento hace que los paquetes eviten las zonas congestionadas utilizando rutas alternativas, con el fin de evitar estas zonas. El problema es que esta política hace que los árboles de congestión se propaguen también a través de todas las rutas alternativas seleccionadas por el algoritmo de encaminamiento adaptativo. Por lo tanto, cuando se utiliza el encaminamiento adaptativo, las soluciones propuestas para tratar el HoL blocking pueden ser no efectivas, como se describe en la siguiente sección.

## III. Planteamiento del Problema

Con el fin de analizar los inconvenientes de las técnicas para tratar los problemas de la congestión propuestas previamente, en un trabajo anterior hemos estudiado la combinación de algoritmos de encaminamientos adaptativos y oblivious con esquemas de colas estáticos en Real Life Fat-Trees (RLFTs) [17]. Básicamente, analizamos la efectividad de los esquemas de colas descritos en la Sección II-C cuando se utilizan algoritmos encaminamientos adaptativos o oblivious. De hecho, observamos que aplicando adaptatividad a los árboles de congestión se generan flujos de tráfico congestionados en toda la red, creando nuevas situaciones de congestión en diferentes puntos de la red. Figura 2 muestra un árbol de congestión generado cuando varios nodos generan un patrón de tráfico "many-to-one", dirigido a un destino "HS".

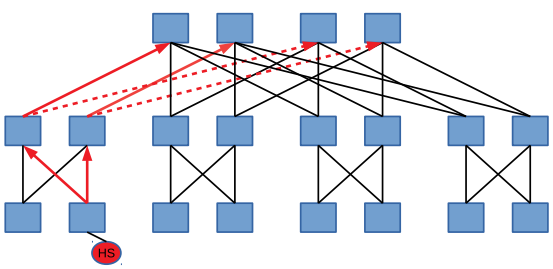



Fig. 2: Situación de congestión en un RLFT de 3 niveles($k = 2$) usando encaminamiento adaptativo. Las flechas rojas muestran el crecimiento de los árboles de congestión que aparecen debido al encaminamiento adaptativo.

En esta situación, los paquetes dirigidos al destino "HS" pueden seleccionar rutas alternativas en dirección ascendente, para evitar los switches en la tercera etapa afectados por los árboles de congestión (flechas rojas en negrita). Como suponemos un patrón de tráfico "muchos a uno", la raíz de la congestión no se mueve a otros puntos finales, y los paquetes dirigidos a "HS" pueden seleccionar rutas alternativas, contribuyendo a generar nuevas ramas del árbol de congestión (flechas rojas discontinuas). Tenga en cuenta que el encaminamiento adaptativo esparce la congestión cuando la congestión es generada por nodos populares que reciben tráfico de muchos nodos fuente. Además, los árboles de congestión no se eliminan, sino que se hacen más fuertes y están más presentes en las rutas de la red, incrementando el HoL blocking y el buffer hogging, degradando aún más el rendimiento de la red. En general, cuando los patrones de tráfico "many-to-one" generan la congestión, los flujos de tráfico adaptados pueden converger y generar situaciones de congestión más fuertes afectando a otras partes de la red.

Por otro lado, la combinación de los algoritmos de encaminamiento adaptativos con los esquemas de colas estáticos (SQSs) implica que se mapearán más destinos por cada VC, ya que este encaminamiento puede seleccionar más rutas que el determinista y por lo tanto habrán más destinos que compartan enlaces. Este incremento de flujos por enlace incrementa la posibilidad de que se genere HoL blocking en los buffers del switch. Además, los SQSs pierden su efectividad para separar flujos en colas y así evitar el HoL blocking y el buffer hogging. Por lo tanto, con el objetivo de ofrecer un algoritmo adaptativo con los beneficios de los SQSs, necesitamos evitar que los algoritmos de encaminamiento esparzan los arboles de congestión por todos los VCs disponibles.

## IV. Aislamiento del Flujo Adaptado

En esta sección, proponemos una nueva técnica llamada Aislamiento del Flujo Adaptado (AFI) que contrarresta las desventajas que generan los algoritmos adaptativos esparciendo la congestión por la red. Pensamos que al limitar la adaptatividad de los paquetes congestionados podemos reducir la congestión que se esparce por rutas alternativas. Básicamente, queremos reducir, tanto como sea posible, las situaciones donde la política de encaminamiento adapte los paquetes para evitar los puntos congestionados en la red. Concretamente, en los buffers del switch AFI, combinamos un esquema de colas estático (SQS) con un VC especial, llamado AFC , el cual se usa para almacenar los paquetes que contribuyen a la congestión. AFI detecta la congestión cuando el algoritmo de encaminamiento adaptativo selecciona una ruta alternativa y marca ese paquete como "adaptado". Aquellos paquetes adaptados son movidos al AFC en el siguiente switch y desde ese momento seleccionan rutas deterministas con el objetivo de que el encaminamiento adaptativo no esparza más este flujo congestionado. Sin embargo, los paquetes "no adaptados" son almacenados en otros VCs siguiendo la política de mapeo especifica del SQSs y son encaminados de forma determinista. Por lo tanto, AFI ajusta de forma dinámica rutas alternativas solo una vez por paquete, si la ruta del paquete atraviesa un área congestionada. En las siguientes secciones describimos los requerimientos de la arquitectura de switch que necesita AFI, como opera y sus detalles de implementación.

### *A. Arquitectura del Switch*

Para simplificar la descripción de la técnica AFI, de ahora en adelante, asumimos el uso de Input Queued (IQ) switches, donde los buffers se colocan en los puertos de entrada. Sin embargo, el uso de AFI no solo está restringido a los IQ switches, sino que

también funciona en otras arquitecturas de switch. La Figura 3 muestra la arquitectura de conmutador asumida.

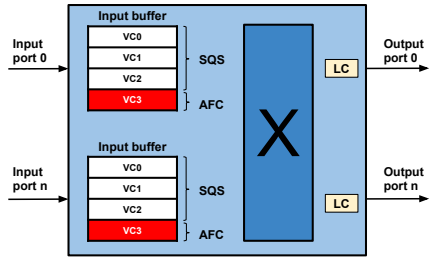



Fig. 3: Arquitectura del switch propuesta para AFI.

Hay al menos dos VCs en el buffer para almacenar paquetes no adaptados, y un AFC por buffer para almacenar los paquetes adaptados. También asumimos que los switches utilizan una política de control de flujo basada en créditos a nivel de VC. Por lo tanto, la lógica del Link Controller (LC) es consciente de los créditos disponibles en el buffer del puerto siguiente. Si la ocupación de un determinado VC a seleccionar por el encaminamiento determinista supera un determinado umbral, esta información es conocida inmediatamente por el LC, ya que realiza un seguimiento de los créditos disponibles por VC. Basándose en esta información, se selecciona un puerto de salida alternativo si el seleccionado de forma determinista tiene el VC congestionado.

Por otro lado, si el algoritmo de encaminamiento no necesita de una ruta alternativa por que los VCs no están lo suficientemente llenos, los paquetes no adaptados son almacenados en los VCs de forma acorde a la política de mapeo de un SQS. De esta forma, nuestra técnica puede ser combinada con los SQS propuestos previamente y es compatible con sus políticas de mapeo especificas siempre que la tecnología de red permita asignar paquetes con algún Service Level o identificador de clase de tráfico [2] Por lo tanto, mientras la congestión no este presente en la red o lo haga de una forma moderada, los SQS son los encargados de separar los flujos de tráfico en VCs. Cuando la congestión se hace persistente y se propaga a otros switches, AFI almacena los paquetes de flujos adaptados en el AFC de los buffers que harán uso de este VC hasta que lleguen al destino. En este caso, los paquetes almacenados en el AFC son encaminados utilizando un algoritmo determinista con el objetivo de no adaptar paquetes que puedan contribuir potencialmente a la congestión. Por otro lado, cuando la ocupación del siguiente buffer de la ruta adaptada baja por debajo del umbral, se deja de adaptar la ruta y se vuelve a utilizar la política de los SQS almacenando los paquetes en los VCs asignados.

Además, podemos asumir que los switches implementan Virtual Output Queues (VOQs), descritos en II-C. En ese caso, los paquetes encaminados hacia el mismo puerto de salida serán almacenados en una VOQ física, evitando así el low-order HoL-blocking. Nótese que cada cada uno de estos paquetes puede estar almacenado en un VC diferente, dependiendo de la política de mapeo del SQS. Por lo tanto, una rama del árbol de congestión puede crecer dentro de una VOQ y, ortogonalmente, a todos los VCs del mismo buffer ya que el control de flujo se realiza a nivel de VC y no a nivel de VOQ. En el caso de los switches basados en la arquitectura VOQ, el buffer hogging puede degradar mucho el rendimiento, ya que todos los VCs del mismo puerto pueden estar llenos de paquetes congestionados. AFI también ayuda a aliviar estas situaciones debido a que condiciona la selección del puerto de salida según un umbral de ocupación del VC en el buffer siguiente, adaptando los paquetes que contribuían a la congestión y almacenándolos solo en el AFC.

### *B. Descripción de Funcionamiento AFI*

La figura 4 muestra como opera AFI cuando aparece la congestión. En esta figura se puede ver una muestra de la red con 3 Switches (A, B y C). En el switch B, la ocupación del VC0 en el puerto de entrada 0 alcanza el umbral de congestión y, desde ese momento, se considera un VC congestionado (Evento #1). Cuando esto sucede, como hemos asumido una política de control de flujo a nivel de VC, el controlador del enlace (LC) del puerto de salida 0 del switch A es consciente de la situación de congestión (Evento #2). Desde ese momento, en el switch A todos los paquetes del VC0 que soliciten ir al switch B por el puerto de salida 0, serán encaminados por un puerto alternativo. Esto sucede cuando un paquete "X" necesita ser encaminado por el puerto de salida 0 del switch A (Evento #3). En ese momento, la lógica de encaminamiento consulta al LC sobre la ocupación del puerto de entrada 0 del switch B. Como el VC está congestionado, en el switch A la lógica de encaminamiento selecciona una ruta alternativa para el paquete "X", encaminándolo por el puerto de salida 1. En este ejemplo, asumimos que se puede encaminar por el puerto de salida 1 del switch A (Evento #4), pero el algoritmo de encaminamiento debe buscar por todos los puertos de salida y seleccionar el más deseable (menor tasa de ocupación del AFC). Una vez el paquete es encaminado sobre la ruta alternativa, el paquete es marcado como "adaptado". Esto se puede implementar mediante la reserva de 1 bit en la cabecera del paquete. De esta forma, cuando llegue al switch C será almacenado en el AFC en el puerto de entrada 0 (Evento #5). Como hemos descrito previamente, una vez el paquete "X" ha sido adaptado, este seguirá una ruta determinista y siempre será almacenado en el AFC.

Por lo tanto, AFI previene el esparcimiento de los flujos adaptados por la congestión sobre rutas alternativas, ya que una vez se detecta la congestión en la red este limita la adaptatividad. Con AFI los flujos de tráfico "adaptados" son encaminados luego de

[2]La terminología Service Level (SL) es empleada por la especificación Infiniband para referirse a las clases o prioridades del tráfico. Otras tecnologías de red, como Intel Omni-Path, usan la terminología Traffic Classes (TC) para referirse a este identificador.

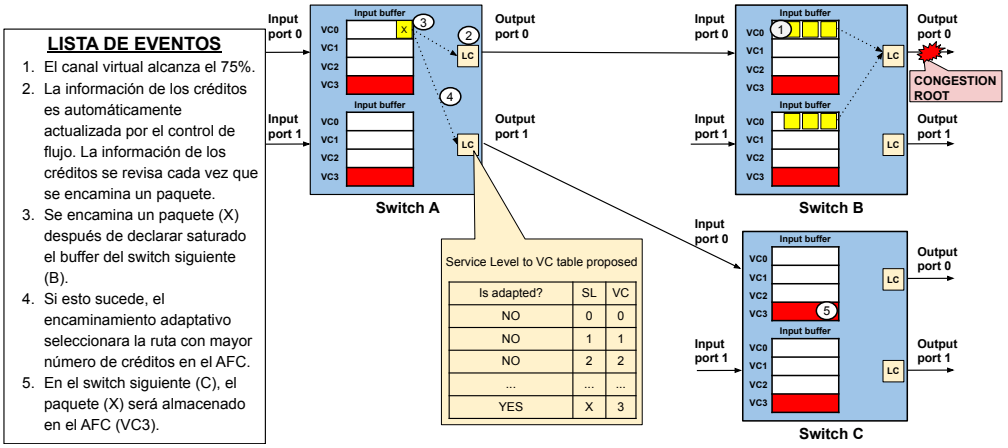



Fig. 4: Ejemplo de funcionamiento de AFI.

forma determinista, mientras que los flujos de tráfico "no adaptados" son almacenados en los VCs de forma acorde a la política del SQS.

### C. Detalles de Implementación

AFI puede ser implementado en tecnologías de red comerciales actuales con un mínimo rediseño y sin añadir una gran complejidad a la arquitectura de los switches. En primer lugar, AFI asume un detector de congestión basado en el nivel de ocupación del VC, el cual puede ser implementado usando la lógica de encaminamiento actual, mirando los LCs que implementan la política del control de flujo. Específicamente, cuando el nivel de ocupación del VC siguiente alcanza el 75 %, nosotros asumimos que los paquetes mapeados a ese VC contribuyen a la congestión o están afectados.

Respecto a la política de mapeo de flujos a VCs, AFI utiliza una tabla en los LCs (ver el LC del Switch A, puerto de salida 1 en la Figura 4) que relaciona los identificadores de la clase de tráfico (SLs or TCs) con los VCs, dependiendo si el paquete ha sido adaptado o no. Los paquetes que han sido adaptados son mapeados al AFC. Esta tabla es similar a las utilizadas en tecnologías de red comerciales. De hecho, algunos de los SQS descritos en la sección II-C han sido implementados utilizando tablas similares. La única diferencia es que se necesita comparar la cabeza del paquete, que contiene el bit de "adaptado", con la tabla del LC para poder mapear el paquete al VC donde será almacenado en el siguiente switch.

Respecto al algoritmo de encaminamiento, nosotros asumimos el uso D-mod-K para Fat-Trees por defecto cuando la red no está congestionada. Una vez que se detecta congestión en la red, entonces se aplica encaminamiento adaptativo y los flujos de tráfico se puede enviar por cualquier puerto de salida disponible. Respecto al arbitraje en el switch, asumimos una política igualitaria entre los SQS y el AFC, donde todos los VCs tienen la misma prioridad.

## V. Evaluación de prestaciones

En esta sección, analizamos el rendimiento de AFI bajo experimentos de simulación, comparando varias propuestas de control de congestión. Hemos llevado a cabo experimentos de simulación en Fat-Trees, bajo tráfico sintético (uniforme y hot-spot), y tráfico basado en la ejecución de aplicaciones reales (mediante archivos de trazas). En esta sección, nosotros describimos los detalles del modelo de simulación usado en nuestra evaluación y comentamos los resultados obtenidos en los experimentos.

### A. Modelo de simulación

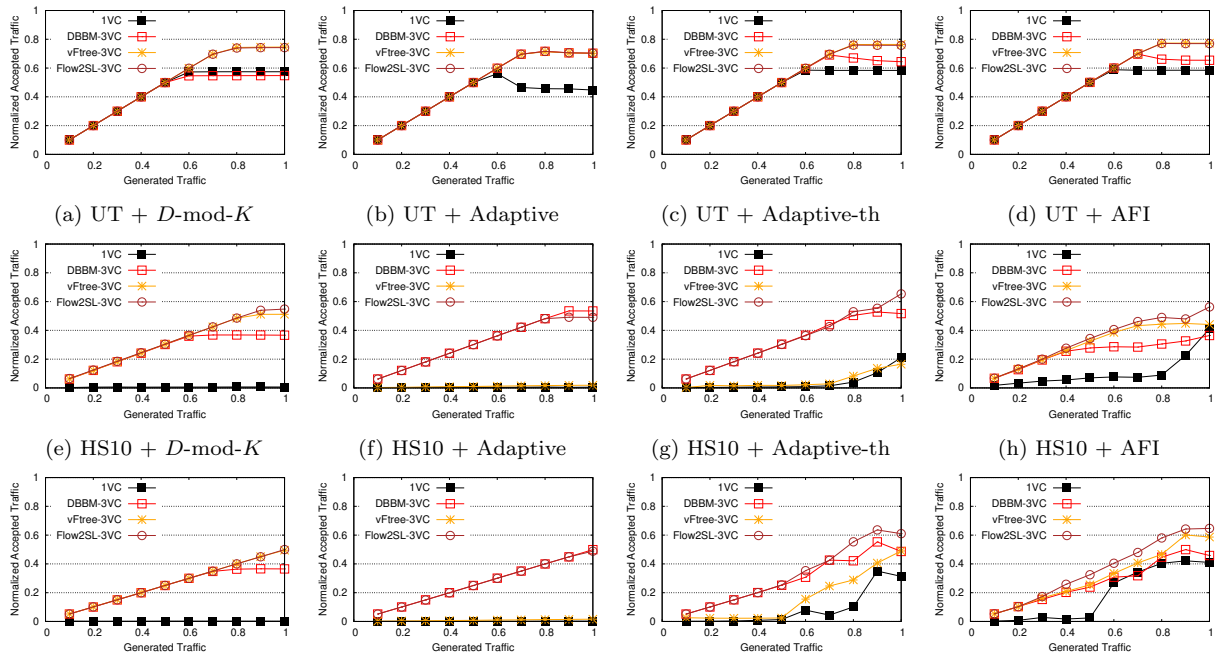
La herramienta de simulación que hemos utilizado en los experimentos es un simulador [22] basado en el framework OMNeT++ [23]. Nosotros hemos implementado en él la técnica AFI para Fat-Trees. La Tabla I muestra las diferentes configuraciones de la topología. Se han modelado dos configuraciones Real Life Fat-Trees (RLFT) con tres etapas, con diferentes tamaños y número de puertos.

TABLA I: Configuración de redes RLFT evaluadas.

| # Config. | Nodos | Puertos | Etapas | Switches |
|---|---|---|---|---|
| 1 | 432 | 12 | 3 | 180 |
| 2 | 11664 | 36 | 3 | 1620 |

Hemos modelado los algoritmos de encaminamiento descritos en [17], es decir, $D$-mod-$K$, adaptativo, adaptativo con umbrales (Adaptive-th) y AFI. Asumimos que AFI usa D-mod-K para rutas deterministas y Adaptive-th cuando adapta flujos de tráfico. Hemos seleccionado un umbral del 75 % para Adaptive-th y AFI. En lo que respecta a la arquitectura de switch, asumimos switches con buffer de entrada (IQ), los cuales pueden ser configurados con VOQs. Asumimos que la política de control de flujo se realiza a nivel de VC, por lo tanto, los flujos de VOQs no tienen control de flujo. Asumimos que el control de flujo está basado en créditos. Hemos modelado AFI para utilizar para poder utilizar cualquier SQSs para comprobar cual funciona mejor junto a nuestra propuesta. Específicamente, hemos modelado Flow2SL, DBBM y vFtree (ver sección II-C) por medio de las siguientes configuraciones de buffer:

- *1VC.* Esta es la configuración de buffer más simple, con sólo 1 VC. Utilizamos este esquema para analizar el la mejora que ofrece AFI cuando no se aplica ningún SQS. En este caso el espacio del buffer se divide en dos VCs (i.e. VC0 and AFC).




(a) UT + $D$-mod-$K$ (b) UT + Adaptive (c) UT + Adaptive-th (d) UT + AFI

(e) HS10 + $D$-mod-$K$ (f) HS10 + Adaptive (g) HS10 + Adaptive-th (h) HS10 + AFI

(i) HS25 + $D$-mod-$K$ (j) HS25 + Adaptive (k) HS25 + Adaptive-th (l) HS25 + AFI

Fig. 5: Productividad Normalizada en función de la carga de Tráfico Generado en un RLFT de 11664-nodos (Config. #2 de la Tabla I), utilizando los algoritmos de encaminamiento $D$-mod-$K$, Adaptativo, Adaptive-th y AFI, los esquemas de colas modelados y patrones de tráfico sintético.

- *3VC*(DBBM-3VC, Flow2SL-3VC y vFtree-3VC). SQSs que usan 3 VCs para separar flujos de tráfico cuando estos son "adaptados". Cuando usamos AFI, el espacio del buffer es divido de igual forma entre los 4 VCs tres VCs para SQS y uno para el AFC.

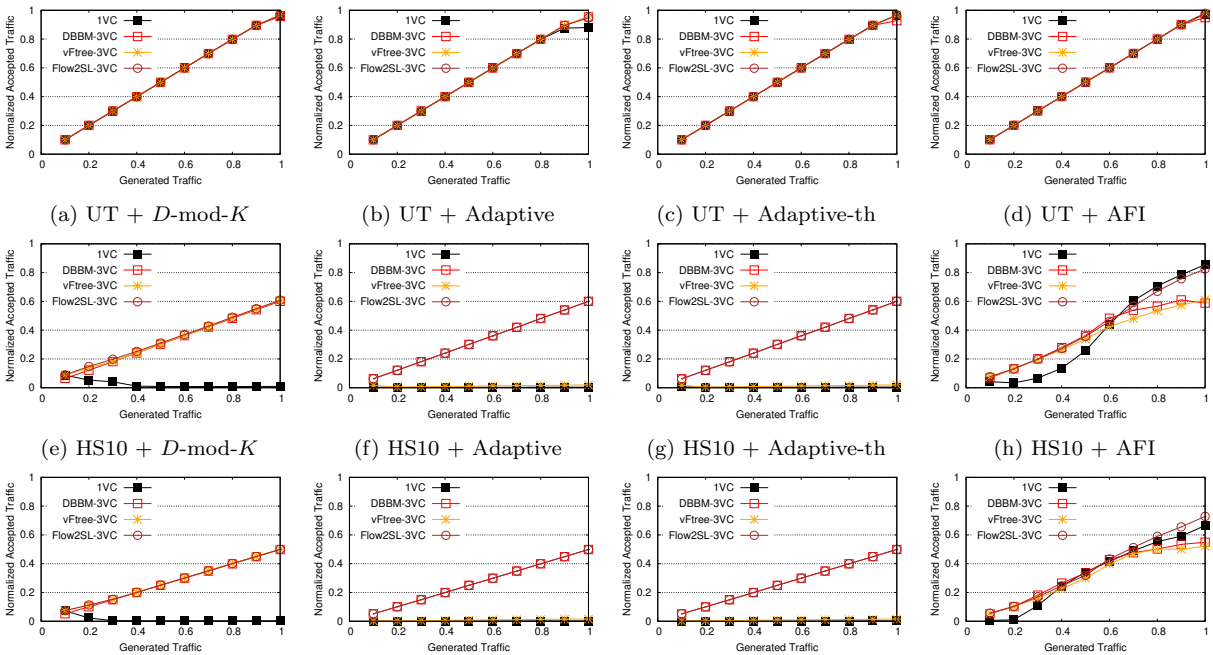
Hemos lanzado experimentos utilizando estas configuraciones en switches IQ con y sin VOQs. El tamaño del buffer de entrada en cada puerto es de 128KB y el tamaño máximo de paquete (MTU) es de 4096 bytes (es decir, se almacenan 32 paquetes por buffer). Asumimos enlaces serie full-duplex con un ancho de banda de 40 Gbps y un retardo de propagación de 6 nanosegundos.

En cuanto a los patrones de tráfico, definimos tres escenarios de tráfico sintéticos diferentes, así como una aplicación basada en trazas reales. El primer patrón de tráfico es el patrón de tráfico uniforme (UT), donde los nodos generan tráfico a destinos aleatorios. El segundo patrón de tráfico (HS10) crea un hot-spot en la red durante todo el tiempo de simulación para modelar un árbol de congestión. Específicamente, en este patrón de tráfico un 10 % de los nodos finales generan paquetes dirigidos a un único destino, mientras que el 90 % de los nodos generan paquetes siguiendo el patrón de tráfico uniforme. El tercer patrón de tráfico (HS25) es también un hot-spot, pero el porcentaje de nodos que generan el hot-spot es del 25 %. Además, se ha modelado el tráfico real gracias al framework de trazas VEF [24]. En particular, utilizamos trazas reales basadas en tráfico de programas MPI obtenidas a partir del test PTRANS, incluido en el benchmark HPCC [25] que genera tráfico entre 432 nodos.

En cuanto a las métricas de rendimiento, medimos el rendimiento normalizado frente a la carga de tráfico (variando la tasa de generación del 0 % al 100 % del ancho de banda máximo del enlace). Finalmente, también proporcionamos el tiempo de ejecución para la aplicación PTRANS.

### *B. Resultados de tráfico sintético*

La Figura 5 y 6 muestran los resultados del experimento para los esquemas de colas modelados con la configuración de red #2 (ver Tabla I), usando arquitecturas de switch sin y con VOQs, respectivamente, y las configuraciones D-mod-k, Adaptative, Adaptive-th y AFI combinados con SQS, bajo patrones de tráfico sintético. La primera fila en la Figura 5 (de la Figura 5a a 5d) muestran los resultados bajo tráfico uniforme(UT). En este caso, el encaminamiento utilizado es indiferente, ya que los periodos de congestión son esporádicos y cortos en el tiempo. Esto se puede observar viendo los resultados de la configuración 1VC que son solo un poco peores cuando se utiliza el encaminamiento Adaptive. La segunda fila de la Figura 5 (de la Figura 5e a 5h) muestra los resultados para un escenario de congestión hotspot (HS10). En este escenario la congestión es fuerte ya que el 10 % de los nodos están generando tráfico direccionado a un único destino. Esto ocasiona que el rendimiento de la configuración 1VC caiga a un valor cercano a 0, y solo reacciona un poco cuando se utilizan las técnicas Adaptive-th o AFI. Lo mismo le sucede a vFtree-3VC cuyo rendimiento mejora significativamente usando AFI (Figura 5h). Además, AFI se sobrepone al problema de diseño de vFtree(la política de mapeo no está pensado para más de dos etapas). Flow2SL y DBBM lidian mejor en este escenario, aunque DBBM no casa muy bien con AFI (obtiene mejores resultados cuando se usa el Adaptativo puro). Flow2SL combinado con AFI logra los




(a) UT + $D$-mod-$K$ (b) UT + Adaptive (c) UT + Adaptive-th (d) UT + AFI

(e) HS10 + $D$-mod-$K$ (f) HS10 + Adaptive (g) HS10 + Adaptive-th (h) HS10 + AFI

(i) HS25 + $D$-mod-$K$ (j) HS25 + Adaptive (k) HS25 + Adaptive-th (l) HS25 + AFI

Fig. 6: Productividad Normalizada en función de la carga de Tráfico Generado en un RLFT de 11664-nodos (Config. #2 de la Tabla I), utilizando los algoritmos de encaminamiento $D$-mod-$K$, Adaptativo, Adaptive-th y AFI, los esquemas de colas modelados y patrones de tráfico sintético. Los switches implementan VOQs

mejores resultados en este escenario. Finalmente, en la tercera fila de la Figura 5 (de la Figura 5i a 5l), la congestión se hace incluso más fuerte, donde el 25 % de los nodos generan tráfico a un único destino. Hay un punto interesante en los resultados obtenidos en las Figuras 5k y 5l. En general los resultados se muestran mejores que para el tráfico HS10.

De la misma forma, aunque los mejores resultados son logrados por Flow2SL combinados con AFI, 1VC y vFtree son los que más mejoran al usar AFI. La Figura 6 muestra los mismos escenarios que antes, pero en este caso los switches son modelados utilizando VOQs con el objetivo de prevenir el low-order HoL blocking. Respecto al patrón de tráfico uniforme, los resultados muestran un rendimiento óptimo independientemente del encaminamiento, ya que la arquitectura del switch previene de forma natural el HoL blocking esporádico. Sin embargo, en los patrones de tráfico HS10 y HS25, el HoL blocking generado es más fuerte debido a que el uso de algoritmos adaptativos es contraproducente para las configuraciones de 1VC y vFtree. En general, cuando se utiliza la técnica AFI el rendimiento de la red mejora significativamente para las técnicas 1VC y vFtree, y en menor medida en Flow2SL y DBBM. Por lo tanto, nuestra conclusión es que AFI contribuye significativamente en estas mejoras de rendimiento.

### C. *Resultados con trazas basadas en aplicaciones reales*

La figura 7 muestra el tiempo de ejecución (en segundos) de los esquemas de cola modelados combinados con los algoritmos de encaminamiento, usando el test PTRANS como carga de trabajo y la configuración de red #1 (ver Tabla I). En general, podemos observar que las técnicas con configuraciones de buffers que usan VOQs obtienen mejores resultados en términos de tiempo de ejecución (más bajo es mejor) que las que no usan VOQs. El mejor resultado es obtenido por 1VC usando VOQs. AFI funciona un poco mejor que el encaminamiento determinista. También podemos ver que el Adaptive-th y D-mod-K logran los mejores resultados, cuando se combina con esquemas de colas, independientemente de la organización del buffer. Sin embargo, hay que tener en cuenta que esto se debe a que la aplicación PTRANS genera un tráfico uniforme con pequeñas ráfagas de congestión. Es decir, el tráfico de la aplicación genera low-order HoL blocking pero no introduce demasiado high-order HoL blocking.

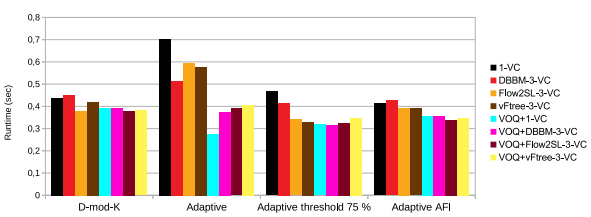



Fig. 7: Tiempo de ejecución de la traza PTRANS.

## VI. Conclusiones

En este trabajo hemos descrito la técnica Adapted-Flow Isolation (AFI), que previene del impacto negativo que origina esparcir flujos congestionados sobre varias rutas cuando se utiliza un algoritmo de encaminamiento adaptativo. Esta propuesta utiliza un mapeo dinámico sobre los canales virtuales (VCs) para marginar los flujos adaptados en un VC especial, mientras el resto de flujos permanecen separados en

otros VCs que siguen una asignación estática de colas. Con esta estrategia, nosotros evitamos que los flujos no adaptados sufran HoL blocking debido al esparcimiento de flujos congestionados por la red. Hemos realizado un extenso conjunto de experimentos para probar nuestra propuesta, modelando grandes topologías Real Life Fat-Trees, bajo tráfico sintético (uniforme y hotspot) y trazas reales basadas en tráfico de programas MPI. El análisis de los resultados obtenidos muestra que AFI mejora los esquemas de cola estáticos en los escenarios fuertemente congestionados. Además, en este trabajo definimos detalles para su implementación tales como el detector de congestión, el motor de encaminamiento, y la tabla de mapeo a VCs. Como trabajo futuro, planeamos probar otros mecanismos para detectar la congestión, aunque el mecanismo utilizado es suficiente para probar la eficiencia de la técnica. Además, la tabla de mapeo de VCs permite explorar otras posibilidades para ampliar AFI utilizando más VCs para otras mejoras.

## Agradecimientos

Este trabajo ha sido co-financiado conjuntamente por el ministerio de Ciencia, Innovación y Universidades español y la Comisión Europea (fondos FEDER) en el marco del proyecto RTI2018-098156-B-C52, y por la Junta de Comunidades de Castilla-La Mancha en el marco de los proyecto PEII-2014-028-P y SBPLY/17/180501/000498. También está cofinanciado por la Universidad de Castilla-La Mancha y FEDER bajo el marco del proyecto 2019-GRIN-27060. Jesús Escudero-Sahuquillo está financiado por la UCLM y la Comisión Europea (fondos FSE), bajo el programa de investigación de la UCLM (Fecha de resolución UCLM: 31/07/2014).

## Referencias

[1] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, Oct 1985.

[2] Crispín Gómez Requena, Francisco Gilabert Villamón, María Engracia Gómez, Pedro López, and José Duato, "Deterministic versus Adaptive Routing in Fat-Trees," in *Proc. of 21th IEEE IPDPS, Long Beach, California, USA*. Mar. 2007, pp. 1–8, IEEE.

[3] Eitan Zahavi, Greg Johnson, Darren J. Kerbyson, and Michael Lang, "Optimized InfiniBand$^{TM}$ fat-tree routing for shift all-to-all communication patterns," *Journal of CCPE*, vol. 22, no. 2, pp. 217–231, 2010.

[4] Eitan Zahavi, Isaac Keslassy, and Avinoam Kolodny, "Distributed adaptive routing convergence to non-blocking DCN routing assignments," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 88–101, 2014.

[5] Patrick Geoffray and Torsten Hoefler, "Adaptive routing strategies for modern high performance networks," in *16th Annual IEEE Symposium on High Performance Interconnects (HOTI 2008), 26-28 August 2008, Stanford, CA, USA*. 2008, pp. 165–172, IEEE Computer Society.

[6] John Kim, William J. Dally, and Dennis Abts, "Adaptive routing in high-radix CLOS network," in *Proc. of the ACM/IEEE Conference on Supercomputing, November 11-17, 2006, Tampa, FL, USA*. 2006, p. 92, ACM Press.

[7] Pedro Javier García, Jose Flich, José Duato, Ian Johnson, Francisco J. Quiles, and Finbar Naven, "Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Arch.," in *Proc of HiPEAC'05, Barcelona, Spain*, pp. 266–285.

[8] M. Jurczyk and T. Schwederski, "Phenomenon of higher order head-of-line blocking in multistage interconnection networks under nonuniform traffic patterns," 1996.

[9] K. Yoshigoe, "Threshold-based exhaustive round-robin for the cicq switch with virtual crosspoint queues," in *2007 IEEE International Conference on Communications*, June 2007, pp. 6325–6329.

[10] William J. Dally and Charles L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.

[11] William Dally, P. Carvey, and L. Dennison, "Architecture of the Avici terabit switch/router," in *6th Hot Interconnects*, 1998, pp. 41–50.

[12] Yuval Tamir and Gregory L. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches," *IEEE Trans. Computers*, vol. 41, no. 6, pp. 725–737, June 1992.

[13] T. Nachiondo, J. Flich, and J. Duato, "Buffer Management Strategies to Reduce HoL Blocking," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 6, pp. 739–753, June 2010.

[14] Jesús Escudero-Sahuquillo, Pedro Javier García, Francisco J. Quiles, José Flich, and José Duato, "OBQA: Smart and cost-efficient queue scheme for Head-of-Line blocking elimination in fat-trees," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1460–1472, 2011.

[15] Wei Lin Guay, Bartosz Bogdanski, Sven-Arne Reinemo, Olav Lysne, and Tor Skeie, "vFtree - A Fat-Tree Routing Algorithm Using Virtual Lanes to Alleviate Congestion," in *In Proc. of 25th IEEE IPDPS 2011, Anchorage, Alaska, USA*. May 2011, pp. 197–208, IEEE.

[16] Jesús Escudero-Sahuquillo, Pedro Javier García, Francisco J. Quiles, Sven-Arne Reinemo, Tor Skeie, Olav Lysne, and José Duato, "A new proposal to deal with congestion in InfiniBand-based fat-trees," *J. Parallel Distrib. Comput.*, vol. 74, no. 1, pp. 1802–1819, 2014.

[17] Jose Rocher-Gonzalez, Jesús Escudero-Sahuquillo, Pedro Javier García, and Francisco J. Quiles, "On the impact of routing algorithms in the effectiveness of queuing schemes in high-performance interconnection networks," in *In Proc. of 25th IEEE HOTI 2017, Santa Clara, CA, USA,*, pp. 65–72.

[18] Eitan Zahavi, "Fat-trees routing and node ordering providing contention free traffic for MPI global collectives," in *In Proc. of 25th IEEE IPDPS Workshops 2011, Anchorage, Alaska, USA,*. pp. 761–770, IEEE.

[19] Mark J. Karol, Michael G. Hluchyj, and Samuel P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Communications*, vol. 35, no. 12, pp. 1347–1356, 1987.

[20] Alejandro Martínez, Raúl Martínez, Francisco José Alfaro, and José L. Sánchez, "A low-cost strategy to provide full qos support in advanced switching networks," *Journal of Systems Architecture*, vol. 53, no. 7, pp. 355–368, 2007.

[21] Pedro Yébenes, German Maglione Mathey, Jesús Escudero-Sahuquillo, Pedro Javier García, and Francisco J. Quiles, "Modeling a switch architecture with virtual output queues and virtual channels in hpc-systems simulators," in *International Conference on High Performance Computing & Simulation, HPCS 2016, Innsbruck, Austria, July 18-22, 2016*, 2016, pp. 380–386.

[22] Pedro Yebenes, Jesús Escudero-Sahuquillo, Pedro Javier García, and Francisco J. Quiles, "Towards modeling interconnection networks of exascale systems with omnet++," in *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013, Belfast, United Kingdom, February 27 - March 1, 2013*, 2013, pp. 203–207.

[23] OpenSim Ltd, "OMNeT++ Discrete Event Simulator," http://omnetpp.org/.

[24] Francisco J. Andujar, Juan A. Villar, Francisco J. Alfaro, José L. Sánchez, and Jesús Escudero-Sahuquillo, "An open-source family of tools to reproduce mpi-based workloads in interconnection network simulators," *The Journal of Supercomputing*, vol. 72, no. 12, pp. 4601–4628, 2016.

[25] "The HPCC benchmark," 2017, http://icl.cs.utk.edu/hpcc/.